IN THE UNITED STATES PATENT and TRADEMARK OFFICE

| | | |
|---|---|---|
| Inventors: | Yigal Mordechai Edery, Nimrod Itzhak Vered, David R. Kroll, Shlomo Touboul | Control No.: Unassigned |
| Patent No.: | 7,058,822 | |
| Issue Date: | June 6, 2006 | |
| Filing Date: | May 17, 2001 | |
| Title: | MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS | |

Mail Stop Ex Parte Reexam
Central Reexamination Unit
Office of Patent Legal Administration
United States Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

**ATTACHMENT TO REQUEST FOR EX-PARTE REEXAMINATION (FORM PTO-SB/57; PTO-1465) PROVIDING INFORMATION ON U.S. PATENT NO. 7,058,822**

Reexamination under 35 U.S.C. §§ 302-307 and 35 C.F.R. § 1.510 is respectfully requested of United States Patent No. 7,058,822 (the "Edery 822 patent"), which was filed on May 17, 2001 and issued on June 6, 2006. The Edery 822 patent is enforceable and reexamination is appropriate under 37 C.F.R. § 1.510(a). The Edery 822 patent currently is being asserted in several patent infringement cases: *Finjan, Inc. v. FireEye, Inc.*, 13-cv-3133 (N.D. Cal.), *Finjan v. Blue Coat Systems, Inc.*, 13-cv-3999 (N.D. Cal.), and *Finjan v. Websense, Inc.*, 13-cv-4398 (N.D. Cal.).

## I. CLAIMS FOR WHICH REEXAMINATION IS REQUESTED

Reexamination is requested of claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 of the Edery 822 patent.

## II. CITATION OF PRIOR ART

The Edery 822 patent was filed on May 17, 2001 as application No. 09/861,229 (the "229 application"). It claims priority to provisional application Ser. No. 60/205,591, which was filed on May 17, 2000.

Requester seeks reexamination of the Edery 822 patent on three separate grounds, each of which raises substantial new questions of patentability ("SNQP"). The bases for the request for reexamination primarily lie in the lack of substantive consideration of the three references, either alone or in combination. Because the request presents these three references in a manner not substantively considered by the Office before, reexamination of the Edery 822 patent is warranted.

Reexamination is first requested in light of U.S. Patent No. 5,983,348 to Ji (the "Ji patent"). The Ji patent was filed on September 10, 1997 and issued on November 9, 1999. Accordingly, the Ji patent is prior art under 35 U.S.C. § 102(e) to the Edery 822 patent. The Ji patent discloses all elements of the Edery 822 patent's claims, specifically claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27. As more thoroughly discussed below, the Ji patent was not substantively considered during the examination of the Edery 822 patent, although it was cited by the examiner.

Reexamination also is requested in light of the combination of the Ji patent and U.S. Patent No. 6,058,482 to Liu (the "Liu patent"). The Liu patent was filed on May 22, 1998. Accordingly, the Ji patent in combination with the Liu patent serves as prior art under 35 U.S.C. § 103(a) to the Edery 822 patent. Together, the combination of the Ji patent and the Liu patent discloses all elements of the Edery 822 patent's claims, specifically claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27. Importantly, the Liu patent was neither cited nor referenced during the prosecution of the Edery 822 patent. Accordingly, the combination of the Ji patent with the Liu patent provides an entirely new ground of invalidity not considered by the examiner.

Reexamination is further requested in light of the combination of the Ji patent and U.S. Patent No. 5,974,549 to Golan (the "Golan patent"). The Golan patent was filed on March 27, 1997. Accordingly, the Golan patent in combination with the Ji patent qualifies as prior art under 35 U.S.C. § 103(a) to the Edery 822 patent. The combination of the Golan patent with the Ji patent was not substantively considered during the examination of the Edery 822 patent, even though the Golan patent itself was disclosed in the specification by the patentee and considered by the examiner. Because this combination of the Ji patent and the Golan patent was not considered by the examiner, this combination provides a third new ground for examination.

## III. STATEMENT POINTING OUT SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

The three bases for reexamination discussed above (the Ji patent alone, the Ji patent in combination with the Liu patent, and the Ji patent in combination with the Golan patent) each establish a substantial new question of patentability of claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 of the Edery 822 patent. These substantial new questions of patentability meet the legal standard for ordering *ex parte* reexamination as set forth in the MPEP § 2216:

> It must first be demonstrated that a patent or printed publication that is relied upon in a proposed rejection presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record during the prosecution of the application that resulted in the patent for which reexamination is requested, and during the prosecution of any other prior proceeding involving the patent for which reexamination is requested.

The Ji patent, the Ji patent in combination with the Liu patent, and the Ji patent in combination with the Golan patent discloses the requisite elements in the claims of the Edery 822 patent for which reexamination is requested. Therefore, the Office should grant this request.

### A. Background of the Edery 822 Patent

Generally, the Edery 822 patent relates to a computer processor-based method that, according to claim 1, includes: (1) a computer receiving downloadable information, (2) if it is determined the downloadable information includes executable code, causing mobile protection code to be communicated to another device, (3) where the determination of whether the downloadable includes executable code involves producing detection indicators which indicate whether a correspondence is detected between the downloadable information characteristics and an executable code characteristic followed by evaluating the relationship between the characteristics to determine if the downloadable information includes executable code.

More specifically, the Edery 822 patent relates to protection systems and methods capable of protecting network accessible devices or processes from malicious operations. The disclosed embodiments provide for determining whether received information from a third party includes executable code. The Edery 822 patent provides for determining, within one or more network servers, whether a received downloadable includes executable code. Additionally, the embodiments provide for a protection engine that operates within one or more network servers,

firewalls, or other network connectable information devices. The protection engine itself includes an information monitor for monitoring the information received by the server. A code detection engine determines whether any of the information received by the server includes executable code.

Furthermore, the embodiments set forth in the Edery 822 patent contemplate "delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables." Edery 822 at Col. 2, lines 43-47. The embodiments further provide for causing the mobile protection code to be executed within the destination, which could include a web browser, in a manner that enables the downloadable operations to be detected, intercepted, or further responded to via the various protected operations. *See e.g.*, Edery 633 at Col. 2 lines 37-55.

The methods disclosed in independent claims 1, 4 and 16 are depicted in Figure 9 of the Edery 822 patent:

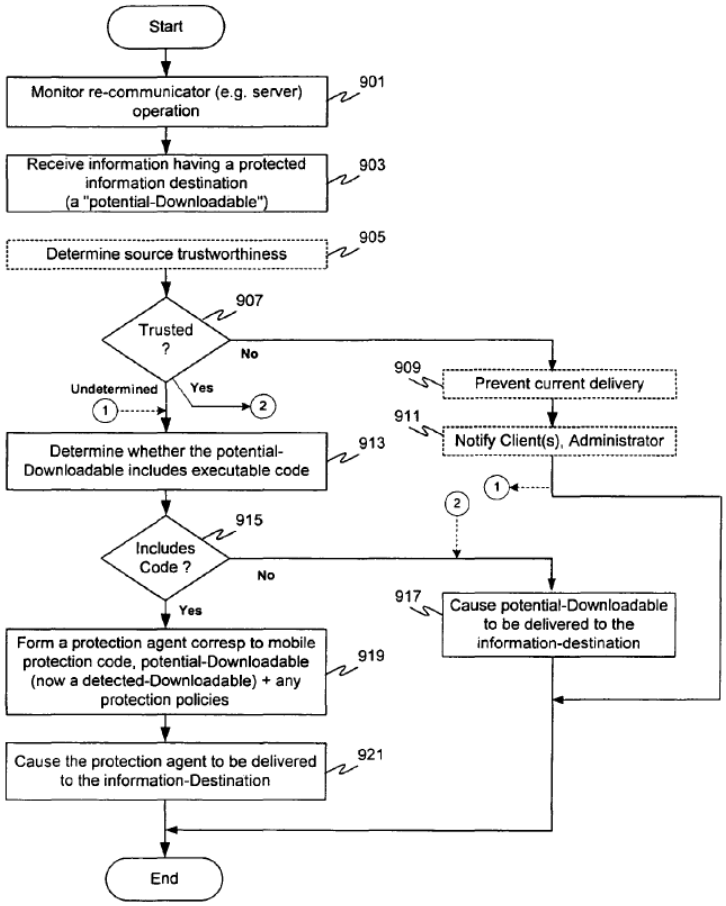



**FIG. 9**

The methods involve a computer receiving downloadable information. Also, the methods provide for the sandboxed package that includes mobile protection code to be received by the computer at a downloadable destination. The computer determines whether the downloadable information includes executable code. If executable code exists, mobile protection code is transmitted from the computer to a third information destination device.

**B. Examination of the Edery 822 Patent**

The Edery 822 patent was prosecuted as the 229 application. On December 7, 2004, pending claims 1-28, 30-59 and 61-75 of the 229 application were rejected under 35 U.S.C. § 101 as being directed toward non-statutory subject matter because the claims recited software alone. The examiner also rejected pending claims 1-7, 16-20, 28-34, 43-51 and 60-76 under 35 U.S.C. § 102(e) as anticipated by the Golan patent. In particular, the examiner found that the Golan patent anticipated the claims of the 229 application because the Golan patent discloses:

> a method, system, and a computer readable storage medium storing computer code for causing a computer to receive downloadable information by a security (information) monitor, determine whether the downloadable information includes executable code as determined by a security monitor (content inspection engine) that is communicatively coupled to the security (information) monitor, and a sandbox (protection agent engine) communicatively coupled to the security monitor (content inspection engine) for causing mobile protection code to be communicated to one information-destination of downloadable information, if the downloadable information is determined to include executable code.

December 7, 2004 Office Action at 4. The examiner also found that the Golan patent's security monitor is "interpreted as acting as both a monitor and an inspector because it examines the downloadable code." *Id*. Therefore, the disclosures in the Golan patent addressed by the examiner anticipated each of the claim elements of the 229 application. Accordingly, at a minimum, the elements from the rejected pending claims existed in the prior art at the time of the alleged invention of the Edery 822 patent.

In the same Office Action, pending claims 8-15, 21-27, 33-42, and 50-59 were objected to due to their dependence on a rejected base claim. However, the examiner stated that these pending claims would likely be allowable if they were rewritten in independent form. Finally, the specification additionally was objected to due to informalities relating to claims of priority.

In response to the December 7, 2004 Office Action, the applicants amended the specification and claims on March 7, 2005. The applicants canceled the pending claims rejected by the examiner, including 1-7, 16-20, 28-34, 43-51 and 70-76. March 7, 2005 Amendment at 3-8. The applicants amended pending claims 8-15, 21-27, 35-42 and 52-69. *Id*. The amendments consisted primarily of adding computer or "processor-based" methods to overcome the section 101 rejections set forth by the examiner. The applicants explained that the amended pending claims were revised to place them in independent form according to the examiner's suggestions.

On April 15, 2005, the applicants filed a supplemental amendment. The specification was amended to modify an incorrect statement relating to the figures, which incorrectly stated that the transfer engine was part of the information being transferred, *e.g*., the sandbox package and/or the not executable potential downloadable. April 15, 2005 Supplemental Amendment at 10. The pending claims 21, 24, and 25 were amended to correct element names. Specifically, the language of the "protection agent engine" and "sandbox package engine" were amended to "packaging engine." *Id*. at 2-5. Additionally, the supplemental amendment changed "protection agent" to "sandbox package." *Id*. at 5.

On June 16, 2005, pending claims 8-15, 21-27, 35-42, 52-59, and 77-80 were allowed. In the Notice of Allowance, the examiner set forth the grounds for allowance of the pending independent claims. Accordingly, this lengthy discussion from the examiner demonstrates the narrow nature of the claims and the examiner's grounds for allowance, which, as detailed in the following section, would not have been allowed upon consideration of the references presented in this request.

For then-pending claim 8 (issued claim 1), the examiner found that the prior art did not teach "performing an analysis on downloadable information, the analysis produces detection indicators indicating whether a correspondence between a downloadable information characteristic and a respective executable code characteristic existed and evaluating those indicators to determine whether the downloadable information included executable code." Notice of Allowance at p. 2.

For then-pending claim 11 (issued claim 4), the examiner found that the prior art did not teach "causing mobile protection code to be communicated to an information destination if the downloadable information is determined to include executable code wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the

mobile protection code and the downloadable information and the sandboxed package is then communicated to an information destination." *Id*.

For then-pending claim 21 (issued claim 9), the examiner found that the prior art did not teach "a content inspection engine that comprises downloadable information analyzers for analyzing downloadable information, [where] each analyzer produces a detection indicator indicating whether a downloadable information characteristic corresponds with an executable code characteristic and an inspection controller couple[d] to the analyzer for determining whether the indicators include that the downloadable information includes executable code." *Id*. at 2-3.

For then-pending claim 24 (issued claim 12), the examiner found that the prior art did not teach "a packaging engine comprises a mobile protection code generator for providing the mobile protection code, a linking engine couple[d] to the mobile protection code generator for forming a sandbox package including the mobile protection code and downloadable information, and a transfer engine for causing the sandbox package to be communicated to an information destination." *Id*. at 3.

For then-pending claims 35 and 52 (issued claims 16 and 28, respectively), the examiner found that the prior art did not teach "causing mobile protection code to be executed by a mobile code executor at a downloadable information destination in that the operations of executable code as a destination, if attempted, will be processed by the mobile protection code and forming a sandboxed package including mobile protection code and downloadable information and causing the sandboxed package to be delivered to a downloadable information destination." *Id*. at 3.

On June 21, 2005, the inventors filed a request to correct inventorship to add Shlomo Touboul as an inventor. The request was granted on August 11, 2005. On June 6, 2006, the 229 application patent issued as the Edery 822 patent.

### C. SNQP 1 - the Ji patent raises a SNQP as to Claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 under 37 CFR 1.510(b)(1)

As discussed above, during prosecution of the Edery 822 patent, the examiner provided a basis for allowance of then-pending claims 8, 11, 24, 35 and 62 (issued claims 1, 4, 12, 16 and 28, respectively). The explained basis for allowance included the proposition that the prior art did not teach creating a sandboxed package that includes mobile protection code. The examiner also found that the prior art did not teach subsequently communicating that sandboxed package,

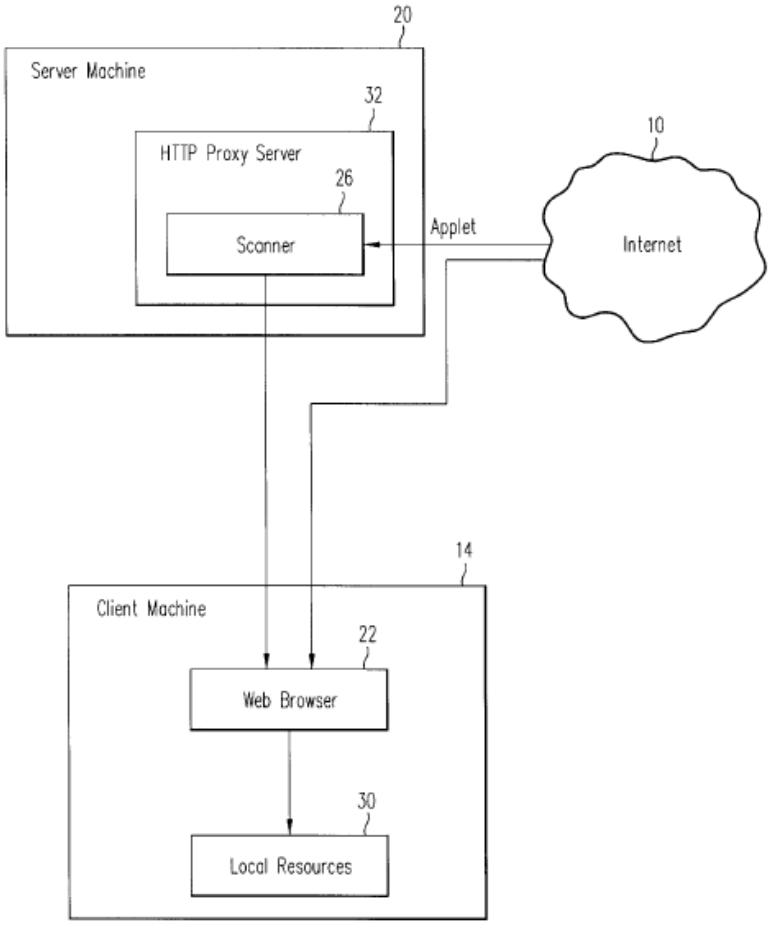
including the downloadable-information and mobile protection code, to an information-destination (client). However, as discussed below, the Ji patent, which was not substantively discussed in the original examination, discloses these elements and thus anticipates the Edery 822 patent.

The Ji patent relates to determining whether a downloadable contains executable code and malware, including teaching communicating mobile protection code from the server to the client. Described more fully in the following paragraphs, the Ji patent performs this analysis in a variety of ways. Accordingly, the Ji patent raises a SNQP because of its disclosure of the elements claimed by the Edery 822 patent and because the Edery 822 patent's prosecution did not include a substantive review of the Ji patent. *See In re Swanson*, 540 F.3d 1368, 1380 (Fed. Cir. 2008) ("the PTO should evaluate the context in which the reference was previously considered and the scope of the prior consideration and determine whether the reference is now being considered for a substantially different purpose").

Although the applicants disclosed the prior art Ji patent in the specification of the Edery 822 patent, they distinguished it in the specification on a different basis than the reasons for allowance cited by the examiner in the Notice of Allowance. Despite this disclosure, the reasons the Ji patent anticipates the Edery 822 patent were not considered by the examiner. As discussed below, the Ji patent teaches creating a sandboxed package including mobile protection code, the downloadable-information and the security policies, where that sandboxed package is subsequently communicated to the intended client destination—the same reasons cited by the examiner in allowing the Edery 822 patent. If the examiner had considered the Ji patent in light of the stated bases for allowance, he would have not allowed the claims to issue.

Like the Edery 822 patent, the Ji patent also relates to a computer processor-based method that includes (1) a computer receiving downloadable information, (2) the computer determining whether the downloadable information includes executable code, (3) if the downloadable information is determined to include executable code, communicating the mobile protection code from the computer to another device and (4) where the determining step of (3) comprises performing analyses of the downloadable-information where the analysis produces detection indicators indicating whether a correspondence is detected between a downloadable information characteristic and at least one executable code characteristic and further evaluating

the detection indicators to determine whether the downloadable information includes executable code. Figure 1 of the Ji patent is shown below:




The Ji patent discloses feature (1), a computer receiving downloadable information (the Applet) at server machine 20, as shown in Figure 1 above and at column 4, lines 55-65. Feature (1) is additionally found in column 3 lines 19-22: "The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server." Accordingly, the Ji patent discloses the first element requiring a computer receiving downloadable-information and therefore raises a SNQP.

The Ji patent also discloses feature (2), "determining whether the downloadable-information includes executable code," by scanning applets and identifying "suspicious instructions" in the downloadable-information for instrumentation . *See* Ji at 3:16-31; 4:66-5:4. The search of suspicious operations in Ji is analogous to determining whether the downloadable-information includes executable code. Accordingly, the Ji patent discloses the feature claimed in

the Edery 822 patent that the computer determines whether the downloadable information includes executable code and therefore raises a SNQP.

The Ji patent also discloses feature (3) which requires that, if the downloadable information is determined to include executable code, the mobile protection code is communicated from the computer to another device, through the disclosure of the monitoring package being delivered to the client. *See* Ji at 3:32-56; 6:38-51. This is particularly evidenced by the Ji patent's disclosure of "pre and post filter monitoring package security policy functions) (sic) are combined with the instrumented applet code in a single JAR (Java archive) file format at the server 32, and downloaded to the web browser 22 in client machine 14." Ji at 6:38-42. Additionally, this is found where "[a]s the applet code is executed, each instrumented instruction is monitored by the web browser using a monitor package which is part of the scanner and delivered to the client side." Ji at 3:35-39. Accordingly, the Ji patent also discloses the feature claimed by Edery 822 that if the downloadable includes executable code, the mobile protection code is transmitted from the computer to another device and therefore raises a SNQP.

The Ji patent further discloses feature (4) including where the determining step of (3) comprises performing analyses of the downloadable-information where the analysis produces detection indicators indicating whether a correspondence is detected between a downloadable information characteristic and at least one executable code characteristic and further evaluating the detection indicators to determine whether the downloadable information includes executable code. The Ji patent discloses feature (4) by disclosing:

> Upon receipt of a particular Java applet, the HTTP proxy server 32, which is software running on server machine 20 and which has associated scanner software 26, then scans the applet and instruments it using an instrumenter 28 which is part of the scanner software 26. (Downloaded non-applets are not scanned.)

Ji at 4:66-5:4. This feature is further disclosed in the Ji patent at 5:16-6:37; 7:8-28. Accordingly, the Ji patent discloses feature (4) of the Edery 822 patent and therefore raises a SNQP.

The Ji patent also discloses all four features of independent claim 4. The first three features of claim 4 are identical to those of claim 1 outlined above, and the explanations set forth above are hereby incorporated by reference. Feature (4) of claim 4 sets forth causing mobile protection code to be communicated comprises forming a sandboxed package including the

mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information destination. The Ji patent discloses this element by disclosing "pre and post filter monitoring package security policy functions) (sic) are combined with the instrumented applet code in a single JAR (Java archive) file format at the server 32, and downloaded to the web browser 22 in client machine 14." Ji at 6:38-42. Therefore, the Ji patent discloses all four features of independent claim 4 and raises a SNQP.

The Ji patent also discloses the three features of independent claim 16 that include (a) receiving at an information re-communicator, downloadable-information, including executable code, (b) causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, (c) wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable- information destination.

The Ji patent discloses feature (a), receiving at a server downloadable-information (the Applet), as shown in Figure 1 above. Feature (a) is additionally found in column 3 lines 19-22: "The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server." Accordingly, the Ji patent discloses the first element requiring a computer receiving downloadable-information and therefore raises a SNQP.

The Ji patent also discloses feature (b) of claim 16 as shown at 4:31-35: "After the code of an applet is downloaded, e.g. via the Internet to a client platform (local computer), an instance of the applet is created in the conventional Java "virtual machine" in the web browser (client) running on that local computer." Feature (b) is additionally found in column 5 lines 8-13:

> The applet is then conventionally interpreted by the web browser 22 and its instructions are executed. The execution is monitored by the monitor package software, also downloaded from scanner 26, in the web browser 22 in accordance with this invention for security purposes.

Accordingly, the Ji patent raises a SNQP as to feature (b) of claim 16.

Finally, the Ji patent discloses feature (c) of claim 16 in the "pre and post filter monitoring package security policy functions) (sic) are combined with the instrumented applet code in a single JAR (Java archive) file format at the server 32, and downloaded to the web

browser 22 in client machine 14". Ji at 6:38-42; *see also* Ji at 6:38-51. Additionally, feature (c) is found where "[a]s the applet code is executed, each instrumented instruction is monitored by the web browser using a monitor package which is part of the scanner and delivered to the client side." Ji at 3:35-39; *see* Ji at 3:32-56. Therefore, Ji discloses all three features of independent claim 16 and raises a SNQP.

To the extent the patentee argues that the Ji patent was discussed during prosecution, the Ji patent was not considered by the examiner for it is disclosed ability of communicating mobile protection code from the server to the client computer. Furthermore, future suggestions by patentee that the Ji patent's disclosure of both static and dynamic Java applet scanning is distinguishable from the Edery 822 patent would be a red herring attempt to distract the examiner with terminology that is not relevant to the identified claims. Therefore, particularly in light of the basis for allowance of claims 4, 12, 16 and 28 and the disclosures from the Ji patent, a substantial new question of patentability is raised.

**D. SNQP 2 – The Ji Patent In View of the Liu Patent Raises a SNQP as to Claims 1, 2, and 3 under 37 CFR 1.510(b)(1)**

The articulated *KSR International Co. v. Teleflex Inc.* obviousness standard[1] dictates that the teachings from the Ji and Liu patents related to the system and methods for protecting network-connectible devices from undesirable downloadable operations and are properly combinable and representative of the obvious body of knowledge well within the grasp of the average practitioner skilled in the art of computer network protection. Therefore, the prior art references, Ji and Liu, are not limited to Java functionality but encompass a broader base of computer technology. One of ordinary skill in the art would be motivated to include the additional disclosures from the Liu patent, because the Liu patent, like the Ji patent, addresses network and computer security, including stopping viruses and other malicious software attacks resulting from the use of network languages like Java. *See* Liu at 1:40-2:58, 4:28-50, 13:1-15; Ji at 1:5-7.

[1] In *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398, 415 (2007), the Supreme court "beg[a]n by rejecting the rigid approach of the Court of Appeals (i.e., requiring satisfaction of the "teaching, suggestion, motivation" (TSM) test) to show an invention would have been obvious (and is therefore unpatentable). Returning to its own nonobviousness cases, the Court held that "the [nonobviousness] analysis need not seek out precise teachings directed to the specific subject matter of the challenged claim, for a court can take account of the inferences and creative steps that a person of ordinary skill in the art would employ." *Id*. at 418.

As discussed in the previous section, the Ji patent discloses all of the elements of the claims of the Edery 822 patent. To the extent the examiner finds that the Ji patent does not disclose the "detection indicators" element found in claims 1, 2, and 3 of the Edery 822 patent, it would have been obvious in light of the Liu patent to use the "detection indicators" disclosed in the Liu patent when performing analyses of the downloaded file. *See* Liu at 1:40-2:58, 3:10-18, 3:31-40,4:28-50, 13:1-15. Therefore, because the combination of the Ji and Liu patents was not considered by the examiner during prosecution, a SNQP exists as to claims 1, 2, and 3 and reexamination is warranted. *See In re Swanson*, 540 F.3d 1368, 1380 (Fed. Cir. 2008) ("the PTO should evaluate the context in which the reference was previously considered and the scope of the prior consideration and determine whether the reference is now being considered for a substantially different purpose").

**E. SNQP 3 – The Ji Patent In View of the Liu Patent Raises a SNQP as to Claims 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 under 37 CFR 1.510(b)(1)**

As with claims 1, 2, and 3 described above, the *KSR* obviousness standard dictates that the teachings from the Ji and Liu patents similarly apply to the claim elements of the Edery 822 patent relating to the "sandboxed package." One of ordinary skill in the art would be motivated to include the "sandbox" and Java architecture because the Liu patent, like the Ji patent, addresses network and computer security, including stopping virus and other malicious software attacks resulting from the use of network languages like Java. Liu at 1:40-2:58, 4:28-50, 13:1-15; Ji at 1:5-7.

Section III.C. above sets forth how the elements disclosed by the Ji patent anticipate the Edery 822 patent. To the extent the examiner requires execution of the "sandboxed package" within a client sandbox and finds that the Ji patent does not disclose the "sandboxed package" claim element, claims 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, and 27 would have been obvious in light of the Liu patent, because the Liu patent discloses client "sandbox" execution for Java related-technology. For example, the disclosure from Liu at 2:19-41 states:

> Given that such ultimately foreign program code is downloaded for local execution, there are inherent security issues that could arise and, therefore, which have been addressed and circumvented in advance within the Java architecture. The Java architecture includes security features that prevent such downloaded programs from interfering with the user's private or non-network resources. Referred to as the "Java sandbox", the Java architecture prevents an untrusted or potentially malicious applet (downloaded to the local end system from a remote

> web server) from reading, writing, or executing private resources, such as the local hard drive. Among other security features, the Java language is a typesafe language, which does not allow pointers to read or write to arbitrary memory locations. In addition, prior to execution of an incoming applet, the applet is run through a Java bytecode verifier, which examines the bytecode for potentially illegal commands, such that only legal applets get executed by the JVM at the local end system. See, e.g., Java Security Whitepaper, available at the Sun web sites java.sun.com and javasoft.com; A Tanenbaum, *Computer Networks* (Prentice-hall, 3d ed. 1996), at 718-20; D. Flanagan, *Java in a Nutshell*, (O'Reilly, 2d ed. 1997), at 7, 139-43.

Therefore, because the combination of the Ji and Liu patents was not considered by the examiner during prosecution, a SNQP exists and reexamination is warranted. *See In re Swanson*, 540 F.3d 1368, 1380 (Fed. Cir. 2008) ("the PTO should evaluate the context in which the reference was previously considered and the scope of the prior consideration and determine whether the reference is now being considered for a substantially different purpose").

### F. SNQP 4 – The Ji Patent In View of the Golan Patent Raises a SNQP as to Claims 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 under 37 CFR 1.510(b)(1)

The *KSR* obviousness standard demonstrates that the teachings from the Ji and Golan patents apply to the claim elements of the Edery 822 patent relating to the "sandboxed package." One of ordinary skill in the art would be motivated to include the specific "sandbox" and Java architecture because the Ji patent, like the Golan patent, addresses network and computer security, including stopping virus and other malicious software attacks resulting from the use of network languages like Java. *See* Ji at 1:1-39; Golan at 1:1-2:8. Importantly, the examiner set forth the bases for which the Golan patent discloses a majority of the claim elements. Also of note, in co-pending Application No. 11/159,455, which matured into U.S. Patent 7,647,633 to Edery *et al*. (the "Edery 633 patent"), the examiner stated during prosecution of the Edery 633 patent that the Golan patent anticipated the then pending claims. The basis upon which the applicants in the prosecution of Edery 633 distinguished the Golan patent was that the Golan patent failed to disclose the "communicating the sandboxed package with the mobile code" element. Application No. 11/159,455 File History May 26, 2010 Amendment at 15-16. Requester has also sought reexamination of the Edery 633 patent as the claims of the Edery 633 and Edery 822 patents parallel one another and raise similar SNQPs. As discussed above in Section III.C., the Ji patent sets forth this allegedly missing element of sandboxing. Accordingly, the Ji patent in combination with the Golan patent raises a SNQP as to claims 4, 5,

6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, and 27, as this combination was not considered by the examiner.

**G. The Edery 822 Patent Is Not Entitled To A Priority Date From The Alleged Parent Continuation-In-Part Patents.**

The Edery 822 patent claims the benefit of (1) an earlier filed provisional application (no. 60/205,591 and (2) purports to be a continuation-in-part of U.S. App. Nos. 09/551,302 and 09/539,667 (now U.S. Patent Nos. 6,480,692 and 6,804,780, respectively). However, the Edery 822 patent cannot claim priority to either of the earlier-filed parent continuations-in-part, because the Edery 822 patent describes and claims new matter not disclosed in the parent applications filed prior to May 17, 2001. As the MPEP explains, determination of the proper priority date is appropriate as part of a request for *ex parte* reexamination:

> The statement applying the prior art may, where appropriate, point out that claims in the patent for which reexamination is requested are entitled only to the filing date of that patent and not supported by an earlier foreign or United States patent application whose filing date is claimed. For example, even where a patent is a continuing application under 35 U.S.C. 120, the effective date of some of the claims could be the filing date of the child application which resulted in the patent, because those claims were not supported in the parent application.

MPEP § 2617.

As illustrated below, although the Edery 822 patent attempts to claim priority to the 6,480,692 and/or 6,804,780 patents, the claims of the Edery 822 patent are "not supported" in those specifications or earlier filed provisional application. *See* U.S. Patent Nos. 6,480,692 and 6,804,780. Accordingly, the applicable priority date for purposes of the invalidity analysis cannot be earlier than May 17, 2001 (the filing date of the 229 application).

The following portions of the Edery 822 patent were all "new matter" in the CIP application:

- FIG. 1a
- FIG. 1b
- FIG. 1c
- FIG. 2
- FIG. 3
- FIG. 4
- FIG. 5
- FIG. 6a
- FIG. 6b
- FIG. 7a
- FIG. 7b
- FIG. 8

- FIG. 9
- FIG. 10a
- FIG. 10b
- FIG. 11
- FIG. 12a
- FIG. 12b
- Col. 1:55 thru Col. 24:45

Consequently, all of the descriptions relating to, among other things, mobile protection code, downloadable-information, information-destination, detection-indicators, executable code characteristic, weighted importance indicators, destination-characteristics, and a sandbox package are all new matter first introduced on May 17, 2001, with the filing of the Edery 822 patent. Similarly, all of the descriptions relating to the aforementioned, including but not limited to determining if the downloadable-information includes executable code, evaluating the significance of various executable code detection indicators, communicating mobile protection code between devices, constructing sandbox packages, constructing sandbox packages that include the downloadable file, mobile protection code and/or security policies are all new matter first introduced on May 17, 2001, the filing date of the Edery 822 patent.

Because the claim scope depends on the newly added material in the application, claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 may only receive the benefit of priority to the 2001 filing date. See, e.g., *Waldemar Link, GmbH & Co. v. Osteonics Corp*., 32 F.3d 556, 558 (Fed. Cir. 1994). Therefore, the Ji patent, Liu patent, and Golan patent all serve as prior art over the Edery 822 patent because their disclosures predate the claimed subject matter of the Edery 822 patent.

## IV. EXPLANATION OF PERTINENCY AND MANNER OF APPLYING CITED PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED UNDER 37 CFR 1.510(B)(2)

Requester submits four proposed grounds corresponding to the four SNQP discussed above as follows:

(A) Claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 are anticipated under 35 U.S.C. § 102(e) in light of the Ji patent (SNQP 1);

(B) Claims 1, 2 and 3 are obvious under 35 U.S.C. § 103 in light of the Ji patent in view of the Liu patent;

(C) Claims 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 are obvious under 35 U.S.C. § 103 in light of the Ji patent in view of the Liu patent; and

(D) Claims 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 are obvious under 35 U.S.C. § 103 in light of the Ji patent in view of the Golan patent (SNQP 4).

**A. The Ji Patent**

The claim chart below details the manner of applying the Ji patent to every claim for which reexamination is requested.

As discussed above, the Edery 822 patent is entitled to a priority date of no earlier than May 17, 2000. The Ji patent was filed on September 10, 1997. Accordingly, the Ji patent anticipates the claims under section 35 U.S.C. § 102(e).

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| 1. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving downloadable-information; | The Ji patent discloses the step of receiving downloadable-information. The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| determining whether the downloadable-information includes executable code; and | The Ji patent discloses the step of determining whether the downloadable-information includes executable code. The Ji patent discloses that the server includes scanning software that scans the downloaded "applet" code and instruments the applet code. Ji at 4:66-5:43. The Ji patent further discloses that non-executable code is not instrumented or scanned, specifically, "[d]ownloaded non-applets are not scanned." Ji at 5:3-4. |
| causing mobile protection code to be communicated to | The Ji patent discloses the step of causing mobile protection code to be communicated to at least one information-destination |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, | of the downloadable-information, if the downloadable-information is determined to include executable code. After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42. |
| wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code. | The Ji patent discloses detecting Java and ActiveX executable code (3:16-22). As the Ji patent explains (and claims in claim 1), the disclosed methods are not limited to Java and ActiveX, but apply to additional executable code types. Ji at 8:23-29. Additionally, the Ji patent discloses performing multiple analyses of the downloaded file to determine if the file contains executable code.<br><br>In the first analysis, Ji discloses that the invention first determines what code (downloaded-information characteristic) is included in the downloaded code (downloaded-information). If the code includes Java or ActiveX code (detection indicator) then that code is of the type (executable code characteristic) identified for further processing and analysis. Ji at 4:66-5:4.<br><br>In the second analysis, Ji discloses instrumenting the applet when "suspicious instructions" (Ji at 5:22) are identified during the scanning process, depicted in Fig. 2 and as described in 5:16-6:37. Accordingly, this second analysis identifies specific applet instructions (downloadable-information characteristics) contained in the downloaded code (downloadable-information) deemed to be "suspicious" (executable code characteristic) as determined by "a predefined set of [insecure] functions." Ji at |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | 5:22-23. Additionally, during the instrumentation process, correspondence between executable code and the code contained within the applet are further indicated by the requirement to download additional Java class files for instrumentation for potentially suspicious instructions:<br>"An applet pre-fetcher component 38 fetches from the Internet 10 all the dependency files required by a Java class file, if they are not already packed into a JAR file. This is important because the goal is to attach the scanner monitor package to a session only once. A Java applet may contain more than one code module, or class file. Heretofore this disclosure has assumed that all the class files are packed in one JAR file and downloaded once. One monitoring package is attached to the JAR file and every instantiation of this package on the client web browser 22 marks a unique session. However, if the class files are not packed together and are downloaded on an as-needed basis during applet execution, multiple instrumentation will occur and multiple instances of the monitoring package for the same session are created on the client. This creates a problem of how to maintain information on session states. To solve this problem, the pre-fetcher 38 pre-fetches the dependency class files during the static scanning of the main applet code module. The dependency class files are (see below) instrumented once, packed together, and delivered to the client."<br>Ji at 7:8-28.<br><br>Thus, the Ji patent discloses determining if the downloaded code is executable and if so, is it a specific type of executable. |
| 2. The method of claim 1, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence. | The Ji patent discloses an indication of a level of downloadable-information characteristic and executable code characteristic correspondence wherein the downloaded code includes Java or ActiveX code (downloadable-information characteristic) because Java and ActiveX applets are executable code characteristics. Ji at 3:17-23; 4:66-5:15. Similarly, as discussed in claim 1 above, the process of instrumenting the executable code contained in the applet shows the correspondence between the downloadable and executable Java instructions. See Ji at 5:16-6:37; 7:8-28. |
| 3. The method of claim 1, wherein the evaluating | The Ji patent inherently discloses assigning a weighted level of importance to at least one of the indicators. For example, if it is |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| includes assigning a weighted level of importance to at least one of the indicators. | Java, then it contains executable code. See Ji at 7:56-59 ("A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold."). |
| 4. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving downloadable-information; | The Ji patent discloses the step of receiving downloadable-information. The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| determining whether the downloadable-information includes executable code; and | The Ji patent discloses the step of determining whether the downloadable-information includes executable code. The Ji patent discloses that the server includes scanning software that scans the downloaded "applet" code and instruments the applet code. Ji at 4:66-5:43. The Ji patent further discloses that non-executable code is not instrumented or scanned, specifically, "[d]ownloaded non-applets are not scanned." Ji at 5:3-4. |
| causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, | The Ji patent discloses the step of causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code. After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42. |
| wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination. | The Ji patent discloses that the instrumented applet communicated from the server to the client's web browser (information-destination) for execution comprises a sandboxed package including mobile protection code (security policy and protection code) and the downloadable-information (instrumented applet). As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 5. The method of claim 4, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information. | The Ji patent discloses a monitoring/security package (mobile protection code) that is executed before the instrumented applet (downloadable-information). As the Ji patent explains, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42; see Fig. 1. And this single JAR archive file contains all of the code necessary for execution of the instrumented applet, including the dependent class files that were similarly instrumented for monitoring during the scanning process. Ji at 7:10-12 ("This [pre-fetching] is important because the goal is to attach the scanner monitor package to a session only once."); see also Ji at 7:8-28.<br><br>Additionally, because the monitoring package (containing the security policy) is designed to monitor the execution of the instrumented applet, the monitoring (and security policies) necessarily should be executed prior to execution of the downloadable for effective security monitoring to occur. Thus, the Ji patent also inherently discloses this limitation. |
| 6. The method of claim 5, wherein the sandboxed | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| package further includes protection policies according to which the mobile protection code is operable. | security (protection) policies necessary to operate the monitoring package. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 7. The method of claim 6, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable-information before the | The Ji patent inherently discloses the possibility of single JAR archive file (sandboxed package) formation wherein the monitoring package software (mobile protection code) is received before instrumented applet (downloadable-information) and the instrumented applet before the specific security (protection) policies.<br><br>First, the monitoring package when previously installed via web page caching may not be refreshed, depending upon browser settings, during subsequent download of a new JAR archive file |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| protection policies. | if the contents of the monitoring package are unchanged. Subsequently, during the download of the new JAR archive file, the instrumented applet code may be downloaded before the portions of the JAR containing the specific security policy code. |
| 8. The method of claim 6, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are specific to the client computer (information-destination) and the user of the client. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 16. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving, at an information re-communicator, downloadable-information, including executable code; and | The Ji patent discloses the step of receiving at a server (information re-communicator) files from the Intranet (downloadable-information) that may include, among other things, applets for execution in a web browser (executable code). The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, | The Ji patent discloses that the web browser (mobile code executor) executes the monitoring package (mobile protection code) contained in the downloaded JAR archive file on the client machine 14 in Fig. 1 (downloadable-information destination). "After the code of an applet is downloaded, e.g. via the Internet to a client platform (local computer), an instance of the applet is created in the conventional Java 'virtual machine' in the Web browser (client) running on that local computer." Ji at 4:31-35.<br><br>And as the applet executes in the browser the instrumented instruction are intercepted by the monitoring package and further action is taken by the monitoring package in accordance with the security policies. "The applet is then conventionally interpreted by the web browser 22 and its instructions are executed. The execution is monitored by the monitor package software, also downloaded from scanner 26, in the web browser 22 in accordance with this invention for security purposes." Ji at 5:8-13. |
| wherein the causing is accomplished by forming a | The Ji patent discloses that the server using the proxy server 32, the scanner 26 (see Fig. 2) creates a single JAR archive file |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable- information destination. | (sandboxed package) containing the applet, monitoring package and security policies 54, and then the server delivers the JAR file to the web browser 22 running on client machine 14 (downloadable-information destination). Ji, Figs. 1 & 2.<br><br>As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 17. The method of claim 16, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies necessary to operate the monitoring package. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 18. A sandboxed package formed according to the | The Ji patent discloses the formation of a single JAR archive file (sandboxed package). The Ji patent explains that, as |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| method of claim 17. | depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 19. The method of claim 17, wherein the forming comprises generating the mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information. | The Ji patent discloses that the single JAR archive file (sandboxed package) is formed by scanning and instrumenting (as describe at 5:16-6:37) the downloaded applet (downloadable-information), then linking the applet to the monitoring package (mobile protection code) and security (protection) policies created by the Security Policy Generator 54 (Fig. 2).<br><br>As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 20. The method of claim 19, wherein the generating of at least one of the mobile | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination. | specific to the client computer (destination). "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 21. The method of claim 20, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are specific to the client computer (destination), including the user, client device or process set for execution. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 22. A sandboxed package formed according to the method of claim 16. | The Ji patent discloses the formation of a single JAR archive file (sandboxed package). The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information. |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 23. The method of claim 16, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator. | The Ji patent discloses communication of the single JAR archive file (sandboxed package) by the HTTP Proxy Server 32 executing the Scanner 26 and Security Policy Generator 54 software through the Server 20 to a client machine 14. Ji at Fig. 1. Accordingly, it is well understood that a communication buffer is standard practice for a server communicating across a network. Thus, the disclosure of a communication buffer is inherent in the Ji patent specification.<br><br>Moreover, the inherency of the communication buffer in any server communicating on a network is established by the Edery 822 patent specification. In Fig. 4 of the Edery 822 patent, the "Buffer" (element 407) shown is not a communication buffer for the "Transfer Engine" (element 406) or specifically for the "sandboxed package" that necessarily must contain the mobile protection code and the downloadable. Instead the "buffer" depicted is specifically placed between the "Information Monitor" (element 401) and "Detection Engine" (element 402). Further, the Edery 822 patent specification repeatedly states that Buffer 407's purpose is "for temporarily storing a received potential-Downloadable." 11:51-52; 11:65-12:1 ("Information monitor 401 monitors potential-Downloadables received by a host server and provides the information via buffer 407 to detection engine 402 or to other system 400 elements."); 14:64-67 (Data fetcher 501 provides for retrieving a potential-Downloadable or portions thereof stored in buffer 407 or parameters from storage 404, and communicates such information or parameters to parser 502. As such, the Edery |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | 822 patent similarly inherently discloses the existence of a "communication buffer" for "communicating the sandboxed package," just as any network server device necessarily requires. |
| 24. The method of claim 16, wherein the re-communicator is at least one of a firewall and a network server. | The Ji patent discloses that the invention software may be installed on a network server. Ji at Fig. 1 (element 20); 3:8-10; 4:59-60; 7:53-55. Additionally, the Ji patent discloses in multiple claims that the re-communicator is a server, e.g. claims 11, 26, 31 and 33. For example, claim 11 states that "[t]he method of claim 1, wherein the computer network includes a server and a client coupled to the server, and wherein the altering takes place at the server, wherein the executing the application program takes place at the client." Ji at 9:1-4.<br><br>Moreover, it was well understood by a person of ordinary skill in the art that any server can be either a network server or a firewall if properly configured. |
| 25. The method of claim 16, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be unaware that the protected package is not a normal downloadable. | The Ji patent discloses the creation of a single JAR archive file (sandboxed package) that contains the downloadable (downloadable-information). A web browser executes Java code in a virtual machine. "After the code of an applet is downloaded, e.g. via the Internet to a client platform (local computer), an instance of the applet is created in the conventional Java 'virtual machine' in the web browser (client) running on that local computer." Ji at 4:30-34. The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code and policy) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Thus, like all JAR files, the JAR archive file executed in the web browser's virtual machine (mobile code executor) is unaware that the protected package is not a normal downloadable. |
| 26. The method of claim 25, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable. | The Ji patent discloses that the single JAR archive file (sandboxed package) is formed by concatenating the downloaded applet (downloadable), monitoring package (mobile protection code) and security (protection) policies into one archive file. The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code and policy) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at |

| Edery 822 Patent Claim Limitations | The Ji Patent |
|---|---|
| | 6:38-42. |
| 27. The method of claim 16, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code. | The Ji patent discloses executing on the client (destination) the instrumented applet in the web browser causes the monitoring package software (mobile protection code) running in the web browser to divert the requested action for evaluation by the monitoring package using the security policies to evaluate the applet request to interface with a local system resource. Ji at 4:30-54.<br><br>After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44. |

### B. The Ji Patent In View Of The Liu Patent

The Ji patent expressly or inherently discloses all of the limitations in claims 1, 2, and 3 as shown above in SNQP 1.

To the extent the Examiner finds that the Ji patent does not disclose the "detection-indicators" claim element, claims 1, 2, and 3 would have been obvious in light of the Liu patent, as shown below. The Liu patent was filed on May 22, 1998. Therefore, the Liu patent is prior art over the claims of the Edery 822 patent, as discussed above.

The claim chart below addresses SNQP 2 and SNQP 3 and details the manner of applying the Ji patent in view of the Liu patent to claims 1, 2, and 3.

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| 1. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving downloadable-information; | The Ji patent discloses the step of receiving downloadable-information. The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| determining whether the downloadable-information includes executable code; and | The Ji patent discloses the step of determining whether the downloadable-information includes executable code. The Ji patent discloses that the server includes scanning software that scans the downloaded "applet" code and instruments the applet code. Ji at 4:66-5:43. The Ji patent further discloses that non-executable code is not instrumented or scanned, specifically, "[d]ownloaded non-applets are not scanned." Ji at 5:3-4. |
| causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, | The Ji patent discloses the step of causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code. After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42. |
| wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code. | The Ji patent discloses detecting Java and ActiveX executable code (3:16-22). As the Ji patent explains (and claims in claim 1), the disclosed methods are not limited to Java and ActiveX, but apply to additional executable code types. Ji at 8:23-29. Additionally, the Ji patent discloses performing multiple analyses of the downloaded file to determine if the file contains executable code.<br><br>In the first analysis, Ji discloses that the invention first determines what code (downloaded-information characteristic) is included in the downloaded code (downloaded-information). If the code includes Java or ActiveX code (detection indicator) then that code is of the type (executable code characteristic) identified for further processing and analysis. Ji at 4:66-5:4.<br><br>In the second analysis, Ji discloses instrumenting the applet when "suspicious instructions" (Ji at 5:22) are identified during the scanning process, depicted in Fig. 2 and as described in 5:16-6:37. Accordingly, this second analysis identifies specific applet instructions (downloadable-information characteristics) contained in the downloaded code (downloadable-information) deemed to be "suspicious" (executable code characteristic) as determined by "a predefined set of [insecure] functions." Ji at 5:22-23. Additionally, during the instrumentation process, correspondence between executable code and the code contained within the applet are further indicated by the requirement to download additional Java class files for instrumentation for potentially suspicious instructions:<br>"An applet pre-fetcher component 38 fetches from the Internet 10 all the dependency files required by a Java class file, if they are not already packed into a JAR file. This is important because the goal is to attach the scanner monitor package to a session only once.<br>A Java applet may contain more than one code module, or class file. Heretofore this disclosure has assumed that all the class files are packed in one JAR file and downloaded once. One monitoring package is attached to the JAR file and every instantiation of this package on the client web browser 22 marks a unique session. However, if the class files are not packed |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | together and are downloaded on an as-needed basis during applet execution, multiple instrumentation will occur and multiple instances of the monitoring package for the same session are created on the client. This creates a problem of how to maintain information on session states. To solve this problem, the pre-fetcher 38 pre-fetches the dependency class files during the static scanning of the main applet code module. The dependency class files are (see below) instrumented once, packed together, and delivered to the client."<br>Ji at 7:8-28.<br><br>Thus, the Ji patent discloses determining if the downloaded code is executable and if so, is it a specific type of executable. |
| 2. The method of claim 1, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence. | As discussed in claim 1 above, it would have been obvious to include the Liu patent's keyword (detection-indicators) in the Ji patent's executable code determination process.<br><br>The Ji patent discloses an indication of a level of downloadable-information characteristic and executable code characteristic correspondence wherein the downloaded code includes Java or ActiveX code (downloadable-information characteristic) because Java and ActiveX applets are executable code characteristics and indicate a definite level of correspondence for the presence of executable code. Ji at 3:17-23; 4:66-5:15. And it would have been obvious to include with the Liu patent's detection of executable code process the further level of correspondence between the keyword attributes (executable code characteristic) and a keyword contained in the downloaded file (downloadable-information characteristic) in conjunction with the executable code determination process. See Liu at 6:19-57. |
| 3. The method of claim 1, wherein the evaluating includes assigning a weighted level of importance to at least one of the indicators. | The Ji patent inherently discloses assigning a weighted level of importance to at least one of the indicators. For example, if it is Java, then it contains executable code. See Ji at 7:56-59 ("A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold."). |

### C. The Ji Patent In View Of The Liu Patent

The Ji patent expressly or inherently discloses all of the limitations in claims 4, 5, 6, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 as shown above in SNQP 1.

To the extent the Examiner finds that the Ji patent does not disclose the "sandboxed package" claim element, claims 4, 5, 6, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 would have been obvious in light of Liu as shown below. The Liu patent was filed on May 22, 1998. Therefore, Liu is prior art over the claims of the Edery 822 patent, as discussed above.

The claim chart below addresses SNQP 2 and SNQP 3 and details the manner of applying the Ji patent in view of the Liu patent to claims 4, 5, 6, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27.

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| 4. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving downloadable-information; | The Ji patent discloses the step of receiving downloadable-information. The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| determining whether the downloadable-information includes executable code; and | The Ji patent discloses the step of determining whether the downloadable-information includes executable code. The Ji patent discloses that the server includes scanning software that scans the downloaded "applet" code and instruments the applet code. Ji at 4:66-5:43. The Ji patent further discloses that non-executable code is not instrumented or scanned, specifically, |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | "[d]ownloaded non-applets are not scanned." Ji at 5:3-4. |
| causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, | The Ji patent discloses the step of causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code. After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42. |
| wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination. | The Ji patent discloses that the instrumented applet communicated from the server to the client's web browser (information-destination) for execution comprises a sandboxed package including mobile protection code (security policy and protection code) and the downloadable-information (instrumented applet). As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 5. The method of claim 4, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information. | The Ji patent discloses a monitoring/security package (mobile protection code) that is executed before the instrumented applet (downloadable-information). As the Ji patent explains, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42; see Fig. 1. And this single JAR archive file contains all of the code necessary for execution of the instrumented applet, including the dependent class files that were similarly instrumented for monitoring during the scanning process. Ji at 7:10-12 ("This [pre-fetching] is important because the goal is to attach the scanner monitor package to a session only once."); see also Ji at |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | 7:8-28.<br><br>Additionally, because the monitoring package (containing the security policy) is designed to monitor the execution of the instrumented applet, the monitoring (and security policies) necessarily should be executed prior to execution of the downloadable for effective security monitoring to occur. Thus, the Ji patent also inherently discloses this limitation. |
| 6. The method of claim 5, wherein the sandboxed package further includes protection policies according to which the mobile protection code is operable. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies necessary to operate the monitoring package. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| 7. The method of claim 6, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable-information before the protection policies. | The Ji patent inherently discloses the possibility of single JAR archive file (sandboxed package) formation wherein the monitoring package software (mobile protection code) is received before instrumented applet (downloadable-information) and the instrumented applet before the specific security (protection) policies.<br><br>First, the monitoring package when previously installed via web page caching may not be refreshed, depending upon browser settings, during subsequent download of a new JAR archive file if the contents of the monitoring package are unchanged. Subsequently, during the download of the new JAR archive file, the instrumented applet code may be downloaded before the portions of the JAR containing the specific security policy code. |
| 8. The method of claim 6, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are specific to the client computer (information-destination) and the user of the client. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 16. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving, at an information re-communicator, downloadable-information, including executable code; and | The Ji patent discloses the step of receiving at a server (information re-communicator) files from the Intranet (downloadable-information) that may include, among other things, applets for execution in a web browser (executable code). The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, | The Ji patent discloses that the web browser (mobile code executor) executes the monitoring package (mobile protection code) contained in the downloaded JAR archive file on the client machine 14 in Fig. 1 (downloadable-information destination). "After the code of an applet is downloaded, e.g. via the Internet to a client platform (local computer), an instance of the applet is created in the conventional Java 'virtual machine' in the Web browser (client) running on that local computer." Ji at 4:31-35.<br><br>And as the applet executes in the browser the instrumented |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | instruction are intercepted by the monitoring package and further action is taken by the monitoring package in accordance with the security policies. "The applet is then conventionally interpreted by the web browser 22 and its instructions are executed. The execution is monitored by the monitor package software, also downloaded from scanner 26, in the web browser 22 in accordance with this invention for security purposes." Ji at 5:8-13. |
| wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable- information destination. | The Ji patent discloses that the server using the proxy server 32, the scanner 26 (see Fig. 2) creates a single JAR archive file (sandboxed package) containing the applet, monitoring package and security policies 54, and then the server delivers the JAR file to the web browser 22 running on client machine 14 (downloadable-information destination). Ji, Figs. 1 & 2.<br><br>As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information. |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 17. The method of claim 16, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies necessary to operate the monitoring package. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 18. A sandboxed package formed according to the method of claim 17. | The Ji patent discloses the formation of a single JAR archive file (sandboxed package). The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 19. The method of claim 17, wherein the forming comprises generating the | The Ji patent discloses that the single JAR archive file (sandboxed package) is formed by scanning and instrumenting (as describe at 5:16-6:37) the downloaded applet |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information. | (downloadable-information), then linking the applet to the monitoring package (mobile protection code) and security (protection) policies created by the Security Policy Generator 54 (Fig. 2).<br><br>As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 20. The method of claim 19, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are specific to the client computer (destination). "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 21. The method of claim 20, wherein the destination- | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process. | security (protection) policies wherein the security policies are specific to the client computer (destination), including the user, client device or process set for execution. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 22. A sandboxed package formed according to the method of claim 16. | The Ji patent discloses the formation of a single JAR archive file (sandboxed package). The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 23. The method of claim 16, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator. | The Ji patent discloses communication of the single JAR archive file (sandboxed package) by the HTTP Proxy Server 32 executing the Scanner 26 and Security Policy Generator 54 software through the Server 20 to a client machine 14. Ji at Fig. 1. Accordingly, it is well understood that a communication buffer is standard practice for a server communicating across a network. Thus, the disclosure of a communication buffer is inherent in the Ji patent specification.<br><br>Moreover, the inherency of the communication buffer in any server communicating on a network is established by the Edery 822 patent specification. In Fig. 4 of the Edery 822 patent, the "Buffer" (element 407) shown is not a communication buffer for the "Transfer Engine" (element 406) or specifically for the "sandboxed package" that necessarily must contain the mobile protection code and the downloadable. Instead the "buffer" depicted is specifically placed between the "Information Monitor" (element 401) and "Detection Engine" (element 402). Further, the Edery 822 patent specification repeatedly states that |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| | Buffer 407's purpose is "for temporarily storing a received potential-Downloadable." 11:51-52; 11:65-12:1 ("Information monitor 401 monitors potential-Downloadables received by a host server and provides the information via buffer 407 to detection engine 402 or to other system 400 elements."); 14:64-67 (Data fetcher 501 provides for retrieving a potential-Downloadable or portions thereof stored in buffer 407 or parameters from storage 404, and communicates such information or parameters to parser 502. As such, the Edery 822 patent similarly inherently discloses the existence of a "communication buffer" for "communicating the sandboxed package," just as any network server device necessarily requires. |
| 24. The method of claim 16, wherein the re-communicator is at least one of a firewall and a network server. | The Ji patent discloses that the invention software may be installed on a network server. Ji at Fig. 1 (element 20); 3:8-10; 4:59-60; 7:53-55. Additionally, the Ji patent discloses in multiple claims that the re-communicator is a server, e.g. claims 11, 26, 31 and 33. For example, claim 11 states that "[t]he method of claim 1, wherein the computer network includes a server and a client coupled to the server, and wherein the altering takes place at the server, wherein the executing the application program takes place at the client." Ji at 9:1-4.<br><br>Moreover, it was well understood by a person of ordinary skill in the art that any server can be either a network server or a firewall if properly configured. |
| 25. The method of claim 16, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be unaware that the protected package is not a normal downloadable. | The Ji patent discloses the creation of a single JAR archive file (sandboxed package) that contains the downloadable (downloadable-information). A web browser executes Java code in a virtual machine. "After the code of an applet is downloaded, e.g. via the Internet to a client platform (local computer), an instance of the applet is created in the conventional Java 'virtual machine' in the web browser (client) running on that local computer." Ji at 4:30-34. The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code and policy) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Thus, like all JAR files, the JAR archive file executed in the web browser's virtual machine (mobile code executor) is unaware that the protected package is not a normal downloadable. |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Liu Patent |
|---|---|
| 26. The method of claim 25, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable. | The Ji patent discloses that the single JAR archive file (sandboxed package) is formed by concatenating the downloaded applet (downloadable), monitoring package (mobile protection code) and security (protection) policies into one archive file. The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code and policy) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42. |
| 27. The method of claim 16, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code. | The Ji patent discloses executing on the client (destination) the instrumented applet in the web browser causes the monitoring package software (mobile protection code) running in the web browser to divert the requested action for evaluation by the monitoring package using the security policies to evaluate the applet request to interface with a local system resource. Ji at 4:30-54.<br><br>After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44. |

### D. The Ji Patent In View Of The Golan Patent

The Ji patent expressly or inherently discloses all of the limitations in claims 4, 5, 6, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 as shown above in SNQP 1.

To the extent the Examiner finds that the Ji patent does not disclose the "sandboxed package" claim element, claims 4, 5, 6, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 would have been obvious in light of Golan patent as shown below. The Golan patent was filed on March 27, 1997. Therefore, the Golan patent is prior art over the claims of the Edery 822 patent, as discussed above.

The claim chart below addresses SNQP 4 and details the manner of applying the Ji patent in view of the Golan patent to claims 4, 5, 6, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27.

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| 4. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving downloadable-information; | The Ji patent discloses the step of receiving downloadable-information. The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| determining whether the downloadable-information includes executable code; and | The Ji patent discloses the step of determining whether the downloadable-information includes executable code. The Ji patent discloses that the server includes scanning software that scans the downloaded "applet" code and instruments the applet code. Ji at 4:66-5:43. The Ji patent further discloses that non-executable code is not instrumented or scanned, specifically, "[d]ownloaded non-applets are not scanned." Ji at 5:3-4. |
| causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code, | The Ji patent discloses the step of causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code. After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42. |
| wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination. | The Ji patent discloses that the instrumented applet communicated from the server to the client's web browser (information-destination) for execution comprises a sandboxed package including mobile protection code (security policy and protection code) and the downloadable-information (instrumented applet). As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 5. The method of claim 4, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information. | The Ji patent discloses a monitoring/security package (mobile protection code) that is executed before the instrumented applet (downloadable-information). As the Ji patent explains, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42; see Fig. 1. And this single JAR archive file contains all of the code necessary for execution of the instrumented applet, including the dependent class files that were similarly instrumented for monitoring during the scanning process. Ji at 7:10-12 ("This [pre-fetching] is important because the goal is to attach the scanner monitor package to a session only once."); see also Ji at 7:8-28.<br><br>Additionally, because the monitoring package (containing the security policy) is designed to monitor the execution of the instrumented applet, the monitoring (and security policies) necessarily should be executed prior to execution of the downloadable for effective security monitoring to occur. Thus, the Ji patent also inherently discloses this limitation. |
| 6. The method of claim 5, wherein the sandboxed package further includes protection policies according | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies necessary to operate the monitoring package. "A security policy defines what functions |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| to which the mobile protection code is operable. | an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 7. The method of claim 6, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable-information before the protection policies. | The Ji patent inherently discloses the possibility of single JAR archive file (sandboxed package) formation wherein the monitoring package software (mobile protection code) is received before instrumented applet (downloadable-information) and the instrumented applet before the specific security (protection) policies.<br><br>First, the monitoring package might have been previously installed via web page caching and not refreshed during subsequent download of a new JAR archive file if the contents of the monitoring package are unchanged. Next, during the download of the new JAR archive file, the instrumented applet |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | code could be downloaded before the portions of the JAR containing the specific security policy code. |
| 8. The method of claim 6, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are specific to the client computer (information-destination) and the user of the client. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 16. A processor-based method, comprising: | The Ji patent discloses the preamble. Ji discloses computer implemented applications executing on a computer network. Specifically, the Abstract discloses a "network scanner for security checking of application programs […] received over the Internet or an Intranet has both static (pre-run time) and |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | dynamic (run time) scanning. […] During run time at the client, the instrumented instructions are thereby monitored for security policy violations, and execution of an instruction is prevented in the event of such a violation."<br><br>Figure 1 discloses a server and client computer device. It is well understood that a computer includes one or more processors for executing software applications. |
| receiving, at an information re-communicator, downloadable-information, including executable code; and | The Ji patent discloses the step of receiving at a server (information re-communicator) files from the Intranet (downloadable-information) that may include, among other things, applets for execution in a web browser (executable code). The Ji patent discloses receiving files "(e.g. Java applets or ActiveX controls)" from the Internet at the server in Fig. 1. See also Ji patent at 3:17-23 ("Thereby in accordance with the invention a scanner (for a virus or other malicious code) provides both static and dynamic scanning for application programs, e.g. Java applets or ActiveX controls. The applets or controls (hereinafter collectively referred to as applets) are conventionally received from e.g. the Internet or an Intranet at a conventional server."). |
| causing mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code, | The Ji patent discloses that the web browser (mobile code executor) executes the monitoring package (mobile protection code) contained in the downloaded JAR archive file on the client machine 14 in Fig. 1 (downloadable-information destination). "After the code of an applet is downloaded, e.g. via the Internet to a client platform (local computer), an instance of the applet is created in the conventional Java 'virtual machine' in the Web browser (client) running on that local computer." Ji at 4:31-35.<br><br>And as the applet executes in the browser the instrumented instruction are intercepted by the monitoring package and further action is taken by the monitoring package in accordance with the security policies. "The applet is then conventionally interpreted by the web browser 22 and its instructions are executed. The execution is monitored by the monitor package software, also downloaded from scanner 26, in the web browser 22 in accordance with this invention for security purposes." Ji at 5:8-13. |
| wherein the causing is accomplished by forming a sandboxed package including the mobile protection code | The Ji patent discloses that the server using the proxy server 32, the scanner 26 (see Fig. 2) creates a single JAR archive file (sandboxed package) containing the applet, monitoring package and security policies 54, and then the server delivers the JAR |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable- information destination. | file to the web browser 22 running on client machine 14 (downloadable-information destination). Ji, Figs. 1 & 2.<br><br>As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 17. The method of claim 16, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies necessary to operate the monitoring package. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 18. A sandboxed package formed according to the method of claim 17. | The Ji patent discloses the formation of a single JAR archive file (sandboxed package). The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 19. The method of claim 17, wherein the forming comprises generating the mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information. | The Ji patent discloses that the single JAR archive file (sandboxed package) is formed by scanning and instrumenting (as describe at 5:16-6:37) the downloaded applet (downloadable-information), then linking the applet to the monitoring package (mobile protection code) and security (protection) policies created by the Security Policy Generator 54 (Fig. 2).<br><br>As Ji discloses "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44.<br><br>The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 20. The method of claim 19, wherein the generating of at least one of the mobile protection code and the protection policies is | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are specific to the client computer (destination). "A security policy defines what functions an applet needs to perform to be |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| conducted in accordance with one or more destination-characteristics of the destination. | considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54.<br><br>"The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 21. The method of claim 20, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process. | The Ji patent discloses that the single JAR file containing the monitoring package and instrumented applet, also contain the security (protection) policies wherein the security policies are specific to the client computer (destination), including the user, client device or process set for execution. "A security policy defines what functions an applet needs to perform to be considered a security risk. Examples of security policies include preventing(1) applets from any file access, or (2) file access in a certain directory, or (3) creating certain Java objects. An applet scanner in accordance With the invention may allow different security policies for different clients, for different users, and for applets from different origins." Ji at 4:47-54. |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution."<br>Ji at 7:41-64. See also Ji at Fig. 2. |
| 22. A sandboxed package formed according to the method of claim 16. | The Ji patent discloses the formation of a single JAR archive file (sandboxed package). The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Moreover, the applet and monitor/security package are executed in a sandbox inside the web browser. "The monitor package contains monitoring functions that are delivered from the server 32 to the client web browser 22 with the instrumental applet and are invoked by the instrumentation code in the applet. The monitor package also creates a unique session upon instantiation. It also contains a security policy checker (supplied by security policy generator component 54) to determine whether the applet being scanned violates the security policy, given the monitoring information.<br><br>The security policy generator component 54 generates the |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | security checker code included in the monitor package, from a set of predefined security policies. Different clients, users, and applets may have different security policies. The security policy generator 54 may run on server machine 20 or another computer. In addition, security policies can be configured by an administrator of the system. A simple security policy is to assign different weights to monitored functions and make sure the security weight of a session does not exceed a preset threshold. A more sophisticated security policy checks the file or resource the applet is trying to access at run time and prompts the user whether to allow the access. Hence the security policy broadly is a state machine to detect security policy violations upon attempted instruction execution." Ji at 7:41-64. See also Ji at Fig. 2. |
| 23. The method of claim 16, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator. | The Ji patent discloses communication of the single JAR archive file (sandboxed package) by the HTTP Proxy Server 32 executing the Scanner 26 and Security Policy Generator 54 software through the Server 20 to a client machine 14. Ji at Fig. 1. Accordingly, it is well understood that a communication buffer is standard practice for a server communicating across a network. Thus, the disclosure of a communication buffer is inherent in the Ji patent specification.<br><br>Moreover, the inherency of the communication buffer in any server communicating on a network is established by the Edery 822 patent specification. In Fig. 4 of the Edery 822 patent, the "Buffer" (element 407) shown is not a communication buffer for the "Transfer Engine" (element 406) or specifically for the "sandboxed package" that necessarily must contain the mobile protection code and the downloadable. Instead the "buffer" depicted is specifically placed between the "Information Monitor" (element 401) and "Detection Engine" (element 402). Further, the Edery 822 patent specification repeatedly states that Buffer 407's purpose is "for temporarily storing a received potential-Downloadable." 11:51-52; 11:65-12:1 ("Information monitor 401 monitors potential-Downloadables received by a host server and provides the information via buffer 407 to detection engine 402 or to other system 400 elements."); 14:64-67 (Data fetcher 501 provides for retrieving a potential-Downloadable or portions thereof stored in buffer 407 or parameters from storage 404, and communicates such information or parameters to parser 502. As such, the Edery 822 patent similarly inherently discloses the existence of a "communication buffer" for "communicating the sandboxed |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| | package," just as any network server device necessarily requires. |
| 24. The method of claim 16, wherein the re-communicator is at least one of a firewall and a network server. | The Ji patent discloses that the invention software may be installed on a network server. Ji at Fig. 1 (element 20); 3:8-10; 4:59-60; 7:53-55. Additionally, the Ji patent discloses in multiple claims that the re-communicator is a server, e.g. claims 11, 26, 31 and 33. For example, claim 11 states that "[t]he method of claim 1, wherein the computer network includes a server and a client coupled to the server, and wherein the altering takes place at the server, wherein the executing the application program takes place at the client." Ji at 9:1-4.<br><br>Moreover, it was well understood by a person of ordinary skill in the art that any server can be either a network server or a firewall if properly configured. |
| 25. The method of claim 16, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be unaware that the protected package is not a normal downloadable. | The Ji patent discloses the creation of a single JAR archive file (sandboxed package) that contains the downloadable (downloadable-information). A web browser executes Java code in a virtual machine. "After the code of an applet is downloaded, e.g. via the Internet to a client platform (local computer), an instance of the applet is created in the conventional Java 'virtual machine' in the web browser (client) running on that local computer." Ji at 4:30-34. The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code and policy) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42.<br><br>Thus, like all JAR files, the JAR archive file executed in the web browser's virtual machine (mobile code executor) is unaware that the protected package is not a normal downloadable. |
| 26. The method of claim 25, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable. | The Ji patent discloses that the single JAR archive file (sandboxed package) is formed by concatenating the downloaded applet (downloadable), monitoring package (mobile protection code) and security (protection) policies into one archive file. The Ji patent explains that, as depicted in Fig. 1, the security monitoring package (mobile protection code and policy) is included with the instrumented downloadable "in a single JAR (Java archive) file format at the server **32**, and downloaded to the web browser **22** in client machine **14**." Ji at 6:38-42. |
| 27. The method of claim 16, | The Ji patent discloses executing on the client (destination) the |

| Edery 822 Patent Claim Limitations | The Ji Patent in View of the Golan Patent |
|---|---|
| wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code. | instrumented applet in the web browser causes the monitoring package software (mobile protection code) running in the web browser to divert the requested action for evaluation by the monitoring package using the security policies to evaluate the applet request to interface with a local system resource. Ji at 4:30-54.<br><br>After the applet code has been instrumented at the server, "[t]he instrumented applet is then downloaded from the server to the client (local computer), at which time the applet code is conventionally interpreted by the client Web browser and it begins to be executed. As the applet code is executed, each instrumented instruction is monitored by the Web browser using a monitor package which is part of the scanner and delivered to the client side. Upon execution, each instrumented instruction is subject to a security check. If the security policy (which has been pre-established) is violated, that particular instruction which violates the security policy is not executed, and instead a report is made and execution continues, if appropriate, with the next instruction." Ji at 3:32-44. |

## V. CONCLUSION

Based on the above remarks, it is respectfully submitted that substantial new questions of patentability have been raised with respect to claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 of the Edery 822 patent. Therefore, reexamination of claims 1, 2, 3, 4, 5, 6, 7, 8, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 and 27 is respectfully requested.

Respectfully submitted,

Dated: October 7, 2013

*/Ryan W. Cobb/*

Ryan W. Cobb
Reg. No. 64,598
Attorney for Requestor

DLA PIPER LLP (US)
401 B Street, Suite 1700
San Diego, CA 92101
ryan.cobb@dlapiper.com
(619) 699-2700
(619) 699-2701

# Electronic Acknowledgement Receipt

| | |
|---|---|
| **EFS ID:** | 17055667 |
| **Application Number:** | 90013017 |
| **International Application Number:** | |
| **Confirmation Number:** | 6388 |
| **Title of Invention:** | MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS |
| **First Named Inventor/Applicant Name:** | Yigal Mordechai Edery |
| **Correspondence Address:** | Ryan W. Cobb<br>DLA Piper LLP (US)<br>401 B Street<br>Suite 1700<br>San Diego CA 92101<br>US 619-699-2635<br>ryan.cobb@dlapiper.com |
| **Filer:** | Ryan Cobb |
| **Filer Authorized By:** | |
| **Attorney Docket Number:** | 382984-000006 |
| **Receipt Date:** | 07-OCT-2013 |
| **Filing Date:** | |
| **Time Stamp:** | 14:42:04 |
| **Application Type:** | Reexam (Third Party) |

## Payment information:

| | |
|---|---|
| Submitted with Payment | yes |
| Payment Type | Deposit Account |
| Payment was successfully received in RAM | $ 12000 |

| | |
|---|---|
| RAM confirmation Number | 851 |
| Deposit Account | [redacted] |
| Authorized User | |

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

## File Listing:

| Document Number | Document Description | File Name | File Size(Bytes)/ Message Digest | Multi Part /.zip | Pages (if appl.) |
|---|---|---|---|---|---|
| 1 | Transmittal of New Application | 01PTOTransmittal822.PDF | 54212<br>6cc3984d8dde58c6cc81f9b21ebd94a874ef18f0 | no | 2 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| 2 | Copy of patent for which reexamination is requested | 02PatentForReexam7058822.pdf | 1771579<br>00472af0111f8ec0073578243be25822a66fc8d2 | no | 25 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| 3 | Receipt of Orig. Ex Parte Request by Third Party | 03RequestForReexam822.pdf | 435886<br>c75b8b89e65b1bcb5f5a2b346fcc2acf1620aeed | no | 67 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| 4 | Reexam - Info Disclosure Statement Filed by 3rd Party | 04IDS822.PDF | 61663<br>5b53bbb5334632436ea1a73fab5ecdc5032653d7 | no | 3 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| 5 | Reexam Miscellaneous Incoming Letter | 05USP5983348.pdf | 652403<br>41fc4327c2c8f971a21930beba391dccb571daab | no | 9 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| 6 | Reexam Miscellaneous Incoming Letter | 06USP6058482.pdf | 1040854<br>09f3068b7cea26f89ebed0831c00740525b14272 | no | 14 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| 7 | Reexam Miscellaneous Incoming Letter | 07USP5974549.pdf | 1447474<br>f9c579325e4188c17f4888ee41860c096d20064f | no | 26 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| 8 | Reexam Certificate of Service | 08CertificateOfService.pdf | 129699<br>a3b2f1241ae0519f9e1d12adf9aa0bd7eee2fa33 | no | 1 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| 9 | Fee Worksheet (SB06) | fee-info.pdf | 29717<br>9aa4d7f19d1abe7952aeba17aeb1c091bc693c40 | no | 2 |
| **Warnings:** | | | | | |
| **Information:** | | | | | |
| | | **Total Files Size (in bytes):** | 5623487 | | |

**This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.**

**New Applications Under 35 U.S.C. 111**
**If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.**

**National Stage of an International Application under 35 U.S.C. 371**
**If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.**

**New International Application Filed with the USPTO as a Receiving Office**
**If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.**